```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load Image
image = cv2.imread('/content/Screenshot 2025-01-30 114305.png')
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Task 1.1: Image Resizing (Interpolation Methods)
linear_resized = cv2.resize(image, (300, 300), interpolation=cv2.INTER_LINEAR)
nearest_resized = cv2.resize(image, (300, 300), interpolation=cv2.INTER_NEAREST)
polynomial_resized = cv2.resize(image, (300, 300), interpolation=cv2.INTER_CUBIC)

# Display resized images
fig, axs = plt.subplots(1, 3, figsize=(15, 5))
axs[0].imshow(cv2.cvtColor(linear_resized, cv2.COLOR_BGR2RGB))
axs[0].set_title('Linear Interpolation')
axs[1].imshow(cv2.cvtColor(nearest_resized, cv2.COLOR_BGR2RGB))
axs[1].set_title('Nearest Neighbors')
axs[2].imshow(cv2.cvtColor(polynomial_resized, cv2.COLOR_BGR2RGB))
axs[2].set_title('Polynomial Interpolation')
for ax in axs:
    ax.axis('off')
plt.show()

# Task 1.2: Image Blurring
# Box Blurring
box_blur = cv2.blur(image, (5, 5))
# Gaussian Blurring
gaussian_blur = cv2.GaussianBlur(image, (5, 5), 0)
# Adaptive Blurring (Using Bilateral Filter)
adaptive_blur = cv2.bilateralFilter(image, 9, 75, 75)

# Display blurred images
fig, axs = plt.subplots(1, 3, figsize=(15, 5))
axs[0].imshow(cv2.cvtColor(box_blur, cv2.COLOR_BGR2RGB))
axs[0].set_title('Box Blurring')
axs[1].imshow(cv2.cvtColor(gaussian_blur, cv2.COLOR_BGR2RGB))
axs[1].set_title('Gaussian Blurring')
axs[2].imshow(cv2.cvtColor(adaptive_blur, cv2.COLOR_BGR2RGB))
axs[2].set_title('Adaptive Blurring')
for ax in axs:
    ax.axis('off')
plt.show()
```

```
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (accuracy_score, precision_score, recall_score,
                             f1_score, confusion_matrix, roc_curve, auc)
import matplotlib.pyplot as plt
import numpy as np

# Load MNIST dataset
digits = load_digits()
X = digits.data  # Feature matrix
y = digits.target  # Target labels

# Split the data into train and test (80-20)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize models
models = {
    "Naive Bayes": GaussianNB(),
    "Random Forest": RandomForestClassifier(random_state=42, n_estimators=100)
}

# Train, evaluate, and compare models
results = {}
kf = KFold(n_splits=5, shuffle=True, random_state=42)

for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Calculate metrics
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='macro')
    recall = recall_score(y_test, y_pred, average='macro')
    f1 = f1_score(y_test, y_pred, average='macro')
    cm = confusion_matrix(y_test, y_pred)

    # Cross-validation score
    cv_scores = cross_val_score(model, X, y, cv=kf, Tscoring='accuracy')

    # Store results
    results[model_name] = {
        "Accuracy": accuracy,
        "Precision": precision,
        "Recall": recall,
        "F1 Score": f1,
        "Confusion Matrix": cm,
        "Cross-Validation Mean Accuracy": np.mean(cv_scores)
    }

# Display results
for model_name, metrics in results.items():
    print(f"Model: {model_name}")
    for metric, value in metrics.items():
        if metric == "Confusion Matrix":
            print(f"{metric}:\n{value}\n")
        else:
            print(f"{metric}: {value}")
    print("-" * 50)
```

```
Model: Naive Bayes
Accuracy: 0.8472222222222222
Precision: 0.8649844547206135
Recall: 0.8476479221745045
F1 Score: 0.8437352605469787
Confusion Matrix:
[[31  0  0  0  0  1  0  1  0  0]
 [ 0 24  0  0  0  0  0  0  3  1]
 [ 0  2 20  0  0  0  1  0 10  0]
 [ 0  0  1 29  0  1  0  0  3  0]
 [ 0  0  0  0 38  0  1  7  0  0]
 [ 0  0  0  1  0 44  1  1  0  0]
 [ 0  0  0  0  1  0 34  0  0  0]
 [ 0  0  0  0  0  1  0 33  0  0]
 [ 0  2  0  0  0  0  0  2 26  0]
 [ 0  1  1  2  0  2  0  4  4 26]]

Cross-Validation Mean Accuracy: 0.8391674404209223
--------------------------------------------------
Model: Random Forest
Accuracy: 0.9722222222222222
Precision: 0.9740424119023985
Recall: 0.9727003722185199
```

```
F1 Score: 0.9732067700933176
Confusion Matrix:
[[32  0  0  0  1  0  0  0  0  0]
 [ 0 28  0  0  0  0  0  0  0  0]
 [ 0  0 33  0  0  0  0  0  0  0]
 [ 0  0  0 32  0  1  0  0  1  0]
 [ 0  0  0  0 46  0  0  0  0  0]
 [ 0  0  0  0  0 45  1  0  0  1]
 [ 0  0  0  0  0  1 34  0  0  0]
 [ 0  0  0  0  0  0  0 33  0  1]
 [ 0  1  0  0  0  0  0  0 29  0]
 [ 0  0  0  0  0  1  0  1  0 38]]

Cross-Validation Mean Accuracy: 0.9755122253172391
--------------------------------------------------
```